입 · 출력 버퍼방식을 이용한 대용량 케이블 점검 시스템 설계 및 구현

Design and Implementation of Large Capacity Cable Checking System using an I/O Buffer Method

양 종 원*
Yang, Jong-Won

ABSTRACT

This paper describes the results on the design and implementation of large capacity cable checking system using I/O buffer method. The I/O buffer module which has feedback loops with input and output buffers is designed with logic gate in the VME board and controlled by MPC860 microprocessor. So this system can check a lot of cable at the same time with less size and less processing time than that of relay matrix method with the A/D converter . The size of the I/O buffer module can be variable according to the number of cable. And any type of cable can be checked even if the pin assignment of cable is changed.

주요기술용어 : I/O Buffer Method, Cable Checker, MPC860, Logic Gate, Node, Link, Microprocessor, Control Signal, VME board

1. 서 론

장, 탈착이 빈번하여 고장이 우려되는 케이블에 대한 점검은 관련 장비 운용의 효율성 측면에서 필수적이다. 특히, 군사장비의 경우 해당 무기체계 관련한 주 · 정비장비 관련 모든 케이블에 대한 주 · 비주기적인 점검이 요구되어 진다. 그러나, 분리 가능한 케이블에 대한 단선, 단락 검사를 수동으로 점검할 경우 많은 시간이 소요되며 자칫 검사자의 실수를 유발할 가능성이 있다. 따라서, 케이블에 대한 자동 검사 장치가 필요하게 되었으며 이를 통해 케이블 점검에 대한 신뢰도 향상 및 점검 소요 시간을 현저히 감소시킬 수 있게 되었다.

그러나, 기존의 아날로그 신호 입 · 출력에 의한 릴레이 매트릭스 방식은 표 1에서처럼 점검하고자하는 핀의 개수만큼 비례하여 A/D 변환기 및 릴레이를 사용하게 되므로 대용량 케이블을 점검할 경우 장비 자체의 크기나 커지게 되므로 대용량 점검에는 적합하

* 국방과학연구소 제2체계개발본부

지 않다.[1][2] 특히, 모듈화가 되어 있지 않아 점검하고자 하는 케이블 개수가 달라질 경우 이에 맞추어 재설계를 해야하는 단점을 가지고 있으며 케이블 연결 정보 변경이 어려워 임의의 케이블에 대한 점검에도 어려움이 있다.

본 논문에서는 다량의 케이블 점검에 적합하도록 입 · 출력 버퍼 방식을 사용한 점검 시스템 구조를 설계하고 이를 H/W로 구현하였다.

입 · 출력 버퍼 방식은 입 · 출력 포트가 같은 노드에서 수행되며 출력된 데이터가 버퍼를 통해 래치(Latch)된 상태에서 궤환을 통해 입력 포트로 재입력되어 출력 노드에 연결된 케이블에 대한 고장 유무를 확인하는 방법이다. 설계된 입 · 출력 버퍼 방식을 단순한 Logic Gate들로 구현할 수 있어 소형화가 가능하고 VME 방식의 보드로 구성할 수 있는 장점으로 인해 모듈화 및 확장성이 용이하여 대용량의 케이블에 대한 점검이 가능하다.

본 시스템은 96핀의 점검이 가능한 입 · 출력 버퍼 처리장치 보드 18개로 구현하였으며 총 1728개의 핀에 대한 동시 점검이 가능하다. 특히, 모듈화된 설계로 인해 점검하고자하는 케이블 용량에 맞는 크기로 사용자가 적절하게 시스템의 크기를 가변적으로 구성할 수 있으며, 케이블 정보에 대한 수정 및 편집이 용이하도록 하여 임의의 케이블에 대한 점검이 가능하도록 하였다.[3]

[표 1] 기존 방법과의 비교

	기존방법	제안방법
방 식	릴레이 매트릭스, A/D 변환기	입 · 출력 버퍼
용 도	소용량 점검 (100핀 이하)	소용량/대용량 점검 (100핀~2000핀 가변)
구 성	단일보드로 구성	주처리장치, I/O버퍼처리장치로 구분
케이블 정보변경	어려움	쉬움(연결정보 DB화)
확장성	확장불가	확장용이 (I/O버퍼처리장치)
모듈화	모듈화 안됨	I/O버퍼처리장치 모듈화

본 논문의 구성은 2장에서 케이블 점검 기본 원리에 대해 설명하고 3장에서는 케이블 점검 시스템의 전체 구성 및 설계 내용에 대하여 보여준다. 4장에서는 입 · 출력 버퍼 방식을 이용하여 구성한 하드웨어를 8비트 기본 모듈로 설계하여 시뮬레이션 수행한 결과를 보여준다. 이를 토대로 32비트 3채널 구조로 확장 설계된 입 · 출력 버퍼 처리장치 구현 및 MPC860 마이크로 프로세서를 이용하여 설계된 주처리장치에 대한 구현 내용과 함께 주요 신호 측정 파형 결과를 5장에서 보이며 6장에서는 결론을 기술하였다.

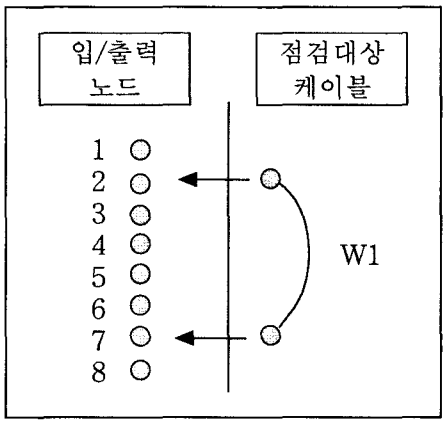
2. 기본 원리

입 · 출력 버퍼 방식을 이해하기 위해서는 먼저 연결 정보를 이용하여 어떻게 단선 · 단락을 표시하는지에 대한 기본 이해가 필요하다.

2.1 연결 정보 표시

점검 시스템이 그림 1과 같이 8개의 노드를 가진 입 · 출력 포트로 구성되고 점검 대상 케이블이 1개의 연결라인만을 가진 선(W1)이며 그 선의 연결은 노드 2와 노드 7을 연결하는 선이라고 가정해 보자.

이 때의 케이블에 대한 연결 정보는 표 2와 같이



[그림 1] 8개 노드에 대한 점검 구성

[표 2] 점검 케이블의 연결정보

wnum[1] = 1	w[1][0] = 1	
wnum[2] = 2	w[2][0] = 2	w[2][1] = 7
wnum[3] = 1	w[3][0] = 3	
wnum[4] = 1	w[4][0] = 4	
wnum[5] = 1	w[5][0] = 5	
wnum[6] = 1	w[6][0] = 6	
wnum[7] = 2	w[7][0] = 7	w[7][1] = 2
wnum[8] = 1	w[8][0] = 8	

작성되는데 wnum[ ]은 연결개수 정보로서 해당 노드에 1을 쓰고 전체 8개 노드에 대한 정보를 모두 읽었을 때에 1이 들어오는 개수를 의미하는데, 이는 해당 노드에 대하여 자기 자신을 포함하여 몇 군데나 연결되는가를 나타낸다. w [ ][ ]는 연결정보로서 해당 노드가 어느 노드와 연결되는가를 나타내는 것으로 표 2는 노드 2와 노드 7이 연결되어야 한다는 것을 표현한 것이다.

2.2 데이터 입 · 출력

먼저 노드 1에 '1'을 쓰고 전체 8개 노드의 데이터를 읽는다. 쓰기 과정과 읽기 과정은 채널이 허용하는 버스 단위로 수행한다. 32비트처리를 가지는 시스템일 경우 32비트가 1개의 채널을 형성하고 입 · 출력을 32비트씩 수행하는데 여기서는 8개 노드를 1개의 채널로 수행한다고 가정 했으므로 8개 노드씩 수행한다. 이 때, 쓰는 과정에서의 데이터는 전체 채널에서 반드시 '1'의 개수가 1개가 되도록 수행한다. 노드 1은 외부에 연결되는 케이블이 없으므로 읽은 데이터는 자기 자신 밖에 없을 것이다. 따라서 이미 정의한 노드 1에 대한 연결개수정보 및 연결정보와 비교하면 wnum[1]= 1, w[1][0]=1 이므로 읽은 데이터가 자기 자신이므로 노드 1을 정상으로 판단한다.

다음에 노드 2에 '1'을 쓰고 전체를 읽는다. 이때는 노드 2와 노드 7이 연결되어 있으므로 노드 7에서도 '1'을 읽게 된다. 즉, 읽은 데이터의 '1'의 개수는 2개이며 노드 2와 노드 7에서 읽은 것이다. 이때 미리 정의한 연결정보와 비교해 보면 wnum[2]=2, w[2][0]=2, w[2][1]=7와 맞게 되므로 이를 정상으로 판단한다. 그러나 만약 미리 정의한 1의 개수가 다를 경우 또는, 1의 위치가 상이할 경우 이를 고장으로 판별한다.

[표 3] 노드별 입출력 결과

번호		1	2	3	4	5	6	7	8
노드	1	1	0	0	0	0	0	0	0
	2	0	1	0	0	0	0	1	0
	3	0	0	1	0	0	0	0	0
	4	0	0	0	1	0	0	0	0
	5	0	0	0	0	1	0	0	0
	6	0	0	0	0	0	1	0	0
	7	0	1	0	0	0	0	1	0
	8	0	0	0	0	0	0	0	1
1의개수		1	2	1	1	1	1	2	1

이러한 과정을 노드 8까지 계속적으로 수행하고 이를 HEX Code로 바꾸어 설명하면 쓴 Data는 0×02이고 읽은 Data는 0×42로 표현되며 이를 비트 별로 분석해 보면 표 3과 같다.

그림 2와 표 4는 정상인 경우와 고장인 경우의 입력 데이터를 비교하였다. 그림 1에서 처럼 노드 2와 노드 7이 연결된 상태를 정상인 경우라고 할 때 노드 2와 노드 6이 연결된 상태는 고장인 경우이다. 노드 2에서 읽은 데이터가 0×42인 경우는 '1'의 개수와 위치가 같으므로 정상이지만, 만약 0×22일 경우 1을 읽은 개수는 같으나 그 위치가 다르므로 고장이며 단선 · 단락으로 표시한다. 노드 6에서 읽은 데이터가 0×22인 경우 '1'의 개수가 많으므로 단락으로 표시하며, 노드 7의 데이터가 0×40일 경우 '1'의 개수가 적으므로 단선으로 표시한다.

Written Data=0x02

8	7	6	5	4	3	2	1
0	0	0	0	0	0	1	0

Read Data=0x42

8	7	6	5	4	3	2	1
0	1	0	0	0	0	1	0

wn[2]=2

wr[2][0]=2,wr[2][1]=7

(a) 정상인 경우

Written Data=0x02

8	7	6	5	4	3	2	1
0	0	0	0	0	0	1	0

Read Data=0x22

8	7	6	5	4	3	2	1
0	0	1	0	0	0	1	0

wn[2]=2

wr[2][0]=2,wr[2][1]=6

(b) 고장인 경우

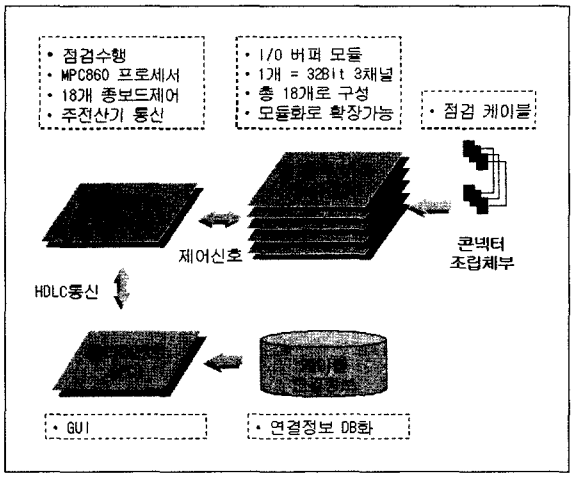
[그림 2] 정상/고장인 경우의 입 · 출력 데이터

[표 4] 정상인 경우와 고장인 경우의 노드별 '1'의 개수와 번호 비교

노드 번호	정상인 경우 (RD=0x42)			고장인 경우 (RD=0x22)		
	1의 개수	1의 번호	판단	1의 개수	1의 번호	판단
1	1	1	정상	1	1	정상
2	2	2, 7	정상	2	2, 6	단선, 단락
3	1	3	정상	1	3	정상
4	1	4	정상	1	4	정상
5	1	5	정상	1	5	정상
6	1	6	정상	2	6, 2	단락
7	2	2, 7	정상	1	7	단선
8	1	8	정상	1	8	정상

3. 시스템 설계 및 구현

3.1 전체 구성

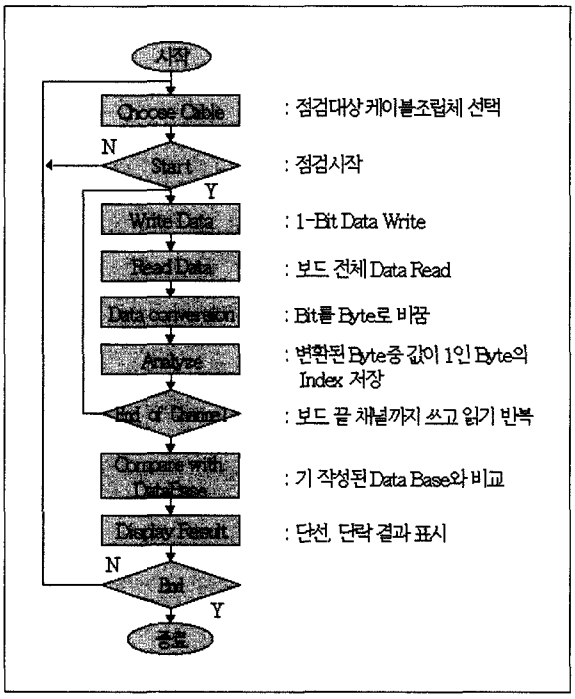


[그림 3] 케이블 점검 시스템 구성

본 논문의 케이블 점검 시스템의 전체 구성은 그림 3과 같이 입 · 출력 버퍼 처리장치, 이를 제어하는 주처리장치(서버), 케이블 연결정보를 포함한 클라이언트(PC)로 구성된다. 기존의 케이블 점검 시스템은 주처리장치와 입 · 출력장치가 하나로 통합되어 있어서 입 · 출력 장치가 허용하는 용량 이상의 점검이 용이하지 않았으며 케이블 연결정보 또한 주처리장치에 포함되고 DB화되지 않아 연결 정보 수정 및 변경이 어려웠다. 본 시스템은 입 · 출력 버퍼 처리 장치 및 연결 정보 DB를 주처리장치와 분리하여 원하는 케이블 용량에 맞도록 입 · 출력 처리장치 및 케이블 연결 정보 DB를 가변적으로 구성이 가능하도록 하였다.

입 · 출력(I/O)버퍼 처리장치는 시뮬레이션을 통해 구현된 입 · 출력 버퍼 모듈을 하드웨어로 구현한 것으로 Logic Gate의 조합으로 구성되어 점검 신호가 점검 케이블로 출력되는 입 · 출력 버퍼 역할을 수행한다. 총 18개의 입 · 출력 버퍼 처리장치로 구성되며 1개의 입 · 출력 버퍼 처리장치는 32비트로 구성된 채널 3개로 구성된다. 주처리장치(서버)는 점검을 총괄하는 메인 프로세서로서 Motorolar사의 MPC860을 사용하여 18개의 입 · 출력 버퍼 처리장치를 제어한다.[8][9][10] 클라이언트는 일반PC로 구성되며 DB화된 케이블 연결 정보를 토대로 정상 유무를 판단한다. DB화된 케이블 연결정보는 TEXT 파일로 수정, 편집이 가능하다.

케이블 점검 시스템의 전체 처리 과정은 그림 4와 같이 구성된다. 여러 케이블 조립체 에서 점검할 케이블 조립체 선택 후에 점검이 수행되며 쓰기/읽기 과정을 전체 보드에 대해 수행 한다. 읽은 데이터 중에서 '1'이 수신된 비트의 개수와 위치를 분석하고 미리 작성된 케이블 연결 정보 및 연결 개수 정보에 의거하여 케이블의 정상 유무를 판단한다.



[그림 4] 케이블 점검 시스템의 처리 과정

3.2 입 · 출력(입 · 출력) 버퍼처리장치의 구성

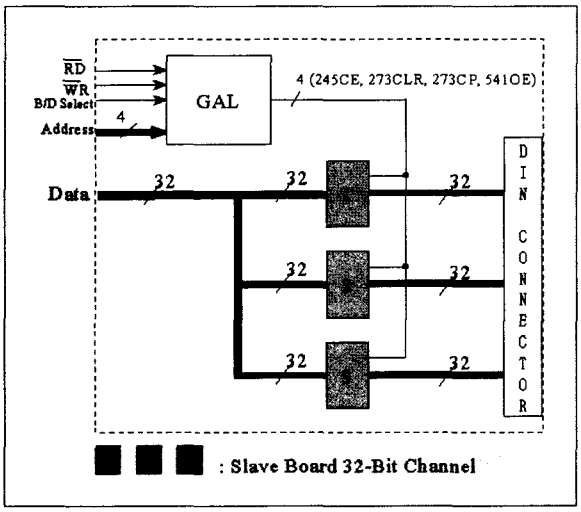
입 · 출력 버퍼 처리장치는 2장에서의 기본 원리를 바탕으로 각 노드를 하나의 버퍼로 구성하고 단선,단락 판별을 위하여 순차적인 제어신호를 발생토록한다. 기존의 방법은 위의 각 노드를 각각 A/D 컨버터로 구성하여 직접 아날로그 신호를 입 · 출력하여 사용하였으나 본 논문은 이를 버퍼로 구성한 것이다.

입 · 출력 버퍼 처리장치는 그림 5와 같이 32비트로 구성된 채널 3개로 구성된다. 이는 주처리장치에서의 메인 프로세서인 MPC860이 32비트 처리 방식을 사용하므로 이를 1개의 채널로 할당한 것이다. 이는 곧 96개(32×2)의 핀 검사를 수행할 수 있다는 것을 말한다. 입 · 출력 버퍼 처리장치는 모듈화되어 있으므로 확장성이 가능하다.

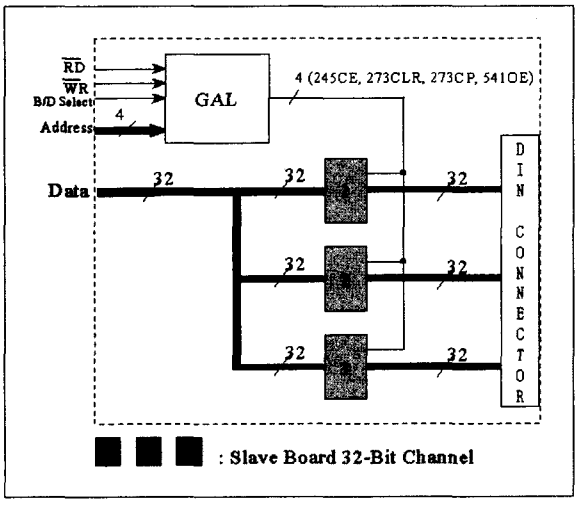
18개로 구성된 입 · 출력 버퍼 처리장치의 경우 총 1728(96×8)개의 핀에 대한 케이블 점검을 할 수 있

다. 제어 신호로써는 읽기 명령 RD, 쓰기 명령 WR, 보드선택 명령 B/D select을 사용하였으며 주소와 데이터에 대해서는 Address 버스와 Data 버스로 구성된다.

1개의 채널은 그림 6과 같이 다시 동일한 구조의 8비트 버퍼 Set 4개로 구성된다. 이는 8비트 기본 모듈 4개를 병렬 연결한 것이다. 이때, 각 입 · 출력 버퍼 처리장치 모듈 내의 Logic Gate에 대한 제어신호와 입 · 출력 버퍼 처리장치를 관리하는 주처리장치에서의 제어 명령을 일관되게 발생시키기 위해서는 이에 따른 제어신호가 필요하다. 이를 위하여 Address 비트 중의 A2~A4를 제어코드로 사용한다. 표 5, 표 6과 같이 A2~A3는 채널 선택을 A4~A5는 제어신호를 발생하는 비트로 할당하고, 이를 표 7과 같이 각 채널에 해당하는 제어신호에 대한 기능을 GAL 소자(P16V8C ; Lattice)를 이용하여 프로그래밍하였다.



[그림 5] I/O버퍼 처리장치 Block Diagram(개략)



[그림 6] I/O버퍼 처리장치 Block Diagram(상세)

[표 5] 제어신호 할당 비트

Address	제어명	제어내용
A5	Control 신호	00 : WR(Write)
A4		01 : RD(Read)
		10 : Reset
A3	Channel 선택	01 : Channel 1
A2		10 : Channel 2
		11 : Channel 3

[표 6] 각 채널에 대한 제어신호 발생

채널	Command	BINARY				제어신호
		A5	A4	A3	A2	십진	
CH1	RD	0	1	0	1	5	OE541,CS245
	Reset	1	0	0	0	8	CLR273
	WR	0	0	0	1	1	CP273,CS245
CH2	RD	0	1	1	0	6	OE541,CS245
	Reset	1	0	0	0	8	CLR273
	WR	0	0	1	0	2	CP273,CS245
CH3	RD	0	1	1	7	7	OE541,CS245
	Reset	1	0	0	0	8	CLR273
	WR	0	0	1	1	3	CP273,CS245

[표 7] 채널1에 대한 GAL 프로그래밍

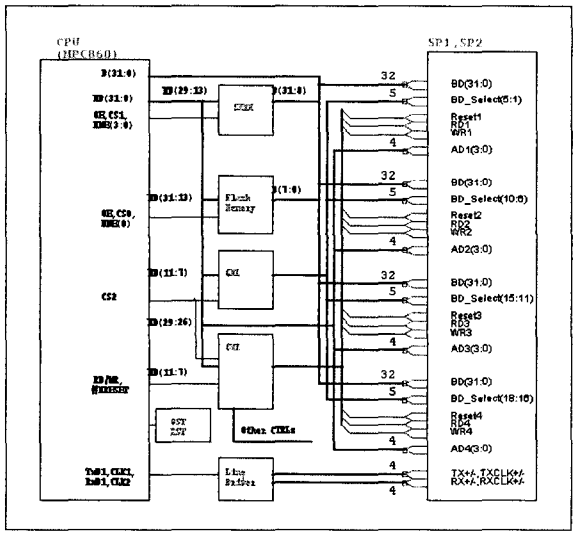
```
MODULE wcabsu64

EQUATIONS ;
 !OE5411  = BDSLCT & (!A5 & A4 & !A3 & A2) ;
 !CLR2731 = BDRESET #
    (BDSLCT & A5 & !A4 & !A3 & !A2 & !WR) ;
 !CP2731  = BDSLCT & (!A5 & !A4 & !A3 & A2) &
    !WR ;
 !CS2451  = BDSLCT & ( (!A5 & !A4 & !A3 & A2) #
    (!A5 & A4 & !A3 & A2) ) ;
END wcabsu64
```

3.3 주처리장치(서버) 구성

18개의 입 · 출력 버퍼 처리장치를 제어하는 주처리장치에 대한 Block Diagram은 그림 7과 같이 MPC860CZP50C1(50MHz) 프로세서를 사용하여 설계하였다.

32비트 Address중에서 하위 16비트는 보드 선택, 상위 16비트는 채널 선택으로 구분하였고 18개 입 · 출력 버퍼 처리장치에 대하여 5개씩 4그룹으로 구분하여 설계하였는데 이는 최대 20개까지 제어가 가능한데 본 시스템에서는 18개까지만 적용하였다. 각 보드별, 채널별 제어신호 발생은 GAL을 이용하여 프로그래밍하였고 입 · 출력 버퍼 처리장치 제어용 메인 프로그램은 메모리 소자를 사용하여 프로그래밍하였다.



[그림 7] 주처리장치 Block Diagram

3.4 클라이언트(PC) 구성

클라이언트는 PC로 구성되어 HDLC 통신을 통하여 주처리장치(서버)에게서 점검 결과를 받아 케이블 연결 정보를 기반으로 단선, 단락을 판단한다. 케이블 연결 정보는 표 2의 기본 형태로 DB화하고 사용자가 케이블에 대한 정보를 쉽게 편집이 가능하여 임의의 케이블에 대한 점검 수행이 가능하도록 하였다.

4. 시뮬레이션

2장에서의 기본 원리를 바탕으로 입 · 출력 버퍼 방식을 이용하여 하드웨어로 설계하기 위하여 8비트 기본 모듈을 시뮬레이션 하였다.

4.1 입 · 출력 버퍼 방식의 동작 특성

입 · 출력 버퍼 방식은 출력과 입력이 같은 노드에서 수행되며 출력된 데이터가 버퍼를 통해 래치(Latch)된 상태에서 궤환을 통해 입력 노드로 재 입력되는 방식이다. 즉, 2장에서의 그림 1의 8개 노드는 각각 8개 버퍼로 구성되며 노드에 연결된 케이블의 정보를 읽기 위해서는 해당 노드에서의 읽기/쓰기 동작을 통한 데이터 흐름 제어가 필요하게 된다.

[표 8] 입 · 출력 버퍼 방식의 동작특성

* 기본 동작은 쓰기(RD), 읽기(WR), 초기화 (Reset)
* 읽기/쓰기 동작이 동일한 버퍼에서 수행된다.
* 제어신호 1 CLK에 쓰기 동작이 발생한다.
* 제어신호 1 CLK에 읽기 동작이 발생한다.
* 출력 신호는 버퍼를 통해 래치(Latch) 된다.
* L'(Low) 출력 신호는 연결 정보 획득을 위해 High Impedance를 발생한다.
* 'H'(High) 출력 신호는 궤환을 통해 자기 자신으로 재입력된다.
* 읽기/쓰기 동작은 채널 단위로 수행된다.
* 쓰기 동작은 전체 채널내에서 'H' 출력이 1개만 발생하도록 한다.

입 · 출력 버퍼방식 설계에 필요한 기본 동작 특성은 표 8과 같다. 기본 동작은 읽기, 쓰기 동작의 반복이며 초기화를 위한 'Reset' 동작을 추가하였다. 기본적인 목적은 자신이 쓰고자 하는 데이터가 자기 자신으로 궤환되어 오는데 이때 자신뿐만 아니라 자신과 연결된 다른 버퍼와의 연결성을 확인하기 위하여 버퍼에 있는 데이터를 동시에 처리를 수행해야 한다는 것이다. 즉, 읽기/쓰기 동작은 단일 클럭으로 수행하며 이는 버스 또는 미리 정한 채널 단위로 수행하게 된다. 이때 쓰고자 하는 데이터는 전체 채널에서 'H' 출력이 1개만 발생하도록 하여 다른 채널과의 간섭을 차단시켜야 하며 이 경우 'H'출력 버퍼 1개를 제외한 나머지 모든 버퍼는 'L'출력을 발생시킨다.[4][5][6][7]

이 때의 'L'출력은 버퍼 출력단에 High Impedance를 발생시켜 주어야 하는데 이는 연결된 다른 포트에서의 간섭을 방지하고자 하는 것이다.

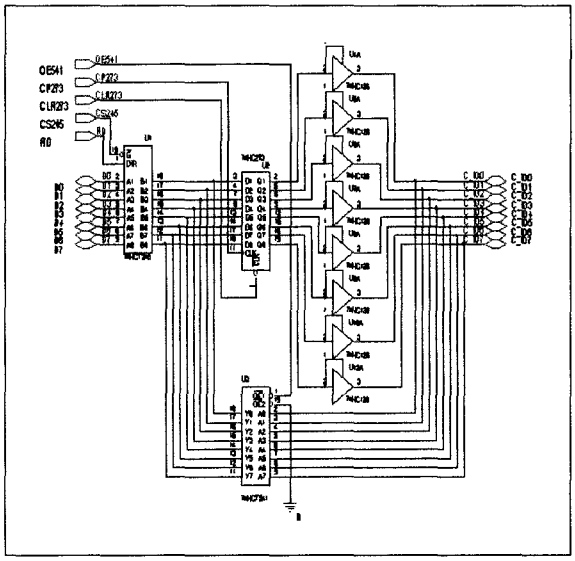
이러한 특성을 갖도록 하드웨어를 설계하고 이에 필요한 제어신호 발생 여부를 시뮬레이션을 통해 확인하고자 한다.

4.2 기본 모듈 시뮬레이션

입 · 출력 버퍼 방식을 이용하여 하드웨어로 구현하기위한 시뮬레이션은 OrCAD Capture 9.0(PSpice 9.0)을 사용하여 수행하였다.

이를 위하여 1채널을 8비트로 구성한 입 · 출력 버퍼 기본 모듈을 그림 8과 같이 4종류의 Logic Gate를 이용하여 설계하였다.

기본 모듈에 사용된 소자로는 74HCT245, 74HC(T)273, 74HC(T)126, 74HCT541이다. 양방향 버퍼(245)를 통해서 쓰기/읽기 과정의 데이터 흐름을 제어하도록 하였고, Latch(273)를 통해 노드에 쓰여진 데이터가 잠시 Latch 되어 원하는 시간에 출력 노드로 데이터를 쓰도록 하였다. 또한, 입력단자와 제어단자를 묶은 3상 버퍼(126)을 통해 'H'일 경우 원하는 출력신호를 발생하고, 'L'일 경우는 'High

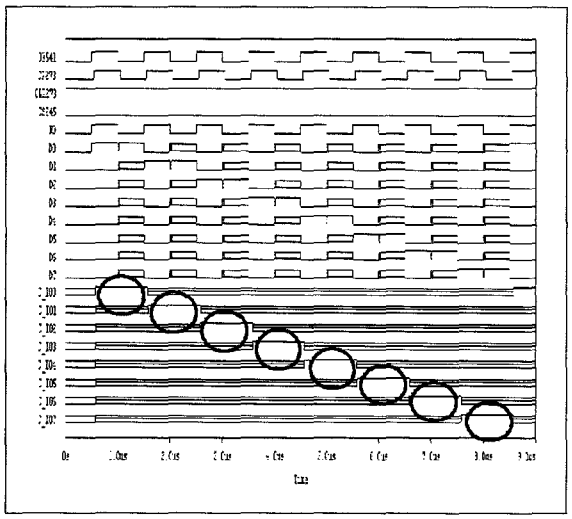


[그림 8] 8비트 처리 입 · 출력 버퍼 기본 모듈

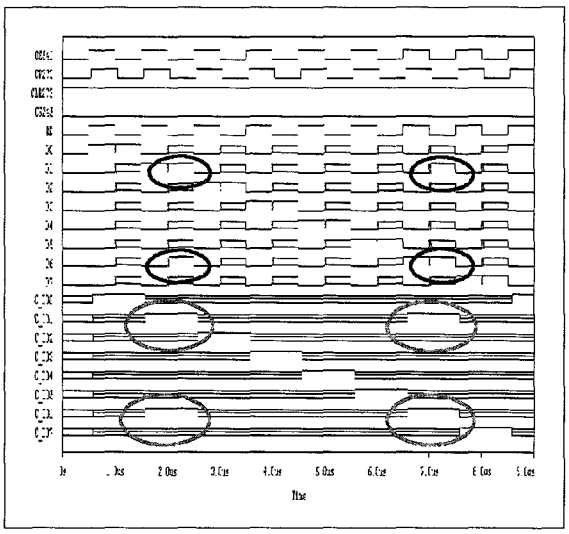
Impedance' 신호를 발생하여 연결된 노드에 대한 읽기 과정에서 'H' 신호 검출을 가능토록 하였고, 단방향 버퍼(541)를 통해 출력 포트에 연결된 신호를 읽기 과정을 통해 궤환되어 재입력되도록 구성하였다.

제어신호는 RD, CS245, CLR27, CP273, OE541의 5가지로 구성하였고 8비트 1채널 기본 모듈에 대하여 표 8을 만족하는 제어 신호 발생을 위하여 표 9와 같은 제어 신호를 구성하였다. 즉, RD가 'H'인 상태에서 CP273을 Enable('H')시킴으로써 쓰기(Write) 과정을 처리하며, RD가 'L'인 상태에서 OE541을 Enable('L')시킴으로써 읽기(Read) 과정을 처리한다.

그림 9와 그림 10은 8비트 기본 모듈에 대해서 bit0에서 bit7까지 8번의 읽기/쓰기 과정의 결과를 보여준다. 그림 9는 출력 노드가 모두 Open 된 상태 즉, 연결된 노드가 하나도 없는 상태이다. 따라서 읽는 데이터 정보가 쓴 데이터와 같게 나오는 것을 알 수 있다. 그림 10은 bit1과 bit6이 연결된 상태에서의 출력결과이다. 두 번째와 일곱 번째 읽기 과정에서 각각 연결된 노드에 해당하는 비트 즉, bit1과 bit6에



[그림 9] 8비트 처리 출력결과(모두 Open 상태)



[그림 10] 8비트 처리 결과(bit1과 bit6이 연결된 상태)

[표 9] 기본 모듈 제어신호

제어신호	WR(Write)	RD(Read)
CP273	1	0
OE541	1	0
CLR273	1	1
CS245	0	0
RD	1	0

해당하는 곳에서 동시에 1' 이 발생됨을 알 수 있다.

$$T_{WR} = t_W + n_{ch} \cdot t_R \quad (1)$$

$$T_n = T_{WR} \cdot n_{bit} \cdot n_{ch} \quad (2)$$

(단, : 1채널 쓰기 시간, : 1채널 읽기 시간, : 1채널 쓰기/읽기 시간, : 총 점검시간)

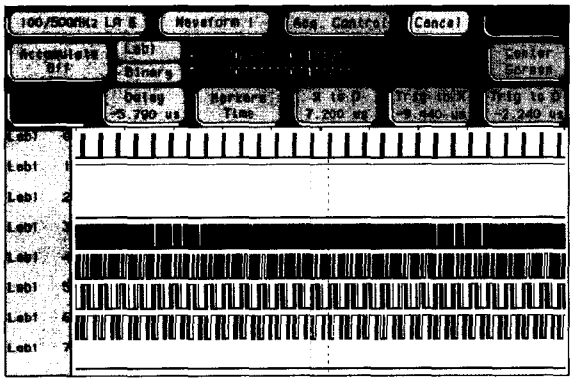
읽기/쓰기 수행 사이클은 식 (1)과 같이 표현되는데

여기서 채널 개수()는 1이고, 사용된 Logic Gate들은 모두(rising time)이 6ns이고, (propagation delay time)이 30ns이하이므로 1비트 읽기/쓰기 사이클은 1us 이내로 수행이 가능하며 식 (2)에서와 같이 비트 폭( )은 8이고, 채널 개수( )인 8비트 1채널에 대한 총 수행시간은 8us 이내로 가능함을 시뮬레이션을 통해 확인하였다.

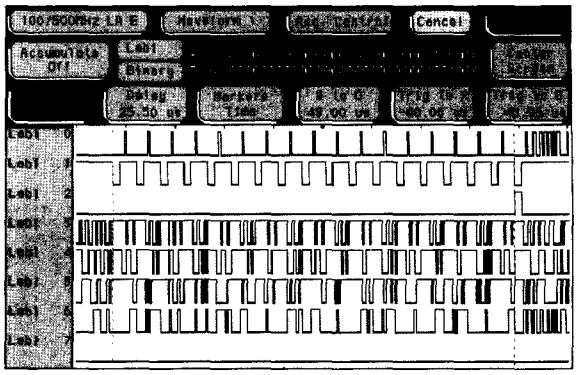
5. 성능 분석

5.1 입 · 출력 버퍼 처리장치 제어신호 발생

구현된 입 · 출력버퍼 처리장치에 대한 제어신호 발생 내역을 확인하기 위하여 Logic Analysis System(HP16500B)를 사용하여 8개의 신호 파형을 분석해 보았다. Lab0 ~ Lab7은 각각 RD, WR, B/D Select, A2, A3, A4, A5, B/D Reset을 나타낸다. 입 · 출력 버퍼 처리장치의 읽기, 쓰기에 대한 기본 동작은 그림 11과 그림 12로 확인한다. 그림 11은 읽기 과정에서의 RD 신호 출력에 대한 제어신호 발생과정을 보인다. RD 동작이므로 WR 신호는 발생하지 않았으며 이에 따른 제어신호가 정상적으로



[그림 11] 읽기 과정의 제어 신호 출력결과



[그림 12] 쓰기/읽기 과정의 18번째 입 · 출력 버퍼 처리장치에 대한 제어신호 출력결과

발생하였다. 그림 12는 18번째 입 · 출력 버퍼 처리장치에서 확인한 것으로써 RD, WR 및 제어신호 신호가 각 보드별로 순차적으로 발생되는 동안 18번째에 해당하는 보드에 이르러서는 해당 B/D Select 신호가 발생되고 이에 해당하는 노드가 쓰기/읽기 과정이 정상적으로 수행하는 것을 확인하였다.

5.2 주처리장치 제어신호 발생

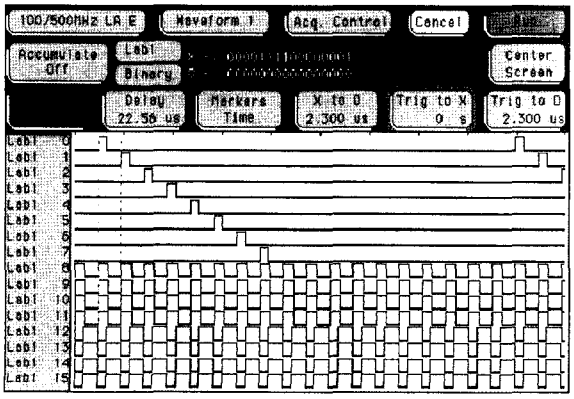
3장의 주처리 장치에서의 기본 동작 특성에 따른 입 · 출력 버퍼 처리장치 제어를 위한 기본 동작은 표 10과 같다. 이는 각 해당 32비트 모듈에 어떠한 방식으로 읽기, 쓰기 동작 신호를 발생시키는 가를 보여주는 것이다. S/W적으로 발생시킨 동작을 실제 H/W로 제대로 동작하는 지의 여부 및 적절한 제어신호가 발생 여부, 그리고 전체 동작 주기 등을 검증하기 위하여 Logic Analysis System(HP16500B)를 사용하여 입 · 출력 버퍼 처리장치 18개 중 상위 8개에 대한 신호 파형을 분석한 결과는 그림 13 ~ 그림 15와 같다.

Lab0 ~ Lab7은 각각 B/D Select 1-8을 나타낸다. 그림 13은 4개의 그룹에 대한 RD1-4(Lab8-11), WR1-4(Lab12-15)를 나타내는데 각 RD, WR 신호 발생에 대하여 B/D Select 신호가 순차적으로 발생하

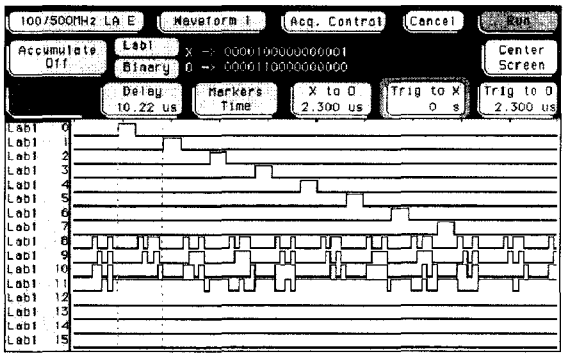
는 것을 볼 수 있다.

[표 10] 주처리장치 메인 프로그래밍

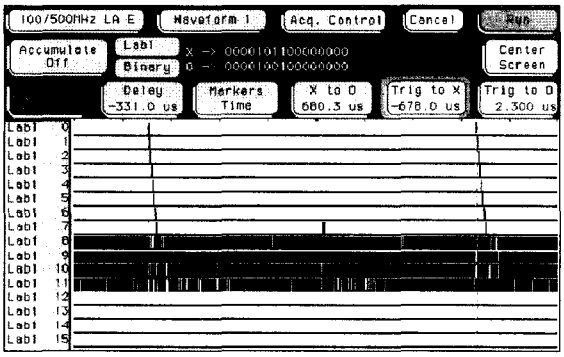
```
void Write_Read(unsigned long *OUT)
{
  for(i=0; i<3; i++){     // 채널개수
     data = 0x00000001;  // 데이터 쓰기 준비
     for(j=0; j<32 ; j++){ // I/O버퍼처리장치 내
                              의 32비트모듈
        *IOTEST = ( (i*32)+(j+1) ) % 16 ;
        *OUT   = data;       // 데이터 쓰기
        ProcWait(300) ;
        for(k=0; k<3 ; k++){  // 채널개수
           for(m=0;m<18;m++)
           databuf.buf32[m] = BD1_IN[k];
           // 18개 I/O 처리장치에 대해 32비트씩
                 데이터 읽기
        } /* FOR 1 END */
        data = data * 2;  // 쓰는 데이터 증가
     } /* FOR 2 END */
     OUT++ ;  // 쓰는 어드레스 증가
   } /* FOR 3 END */
} /* End of void Write_Read() */
```



[그림 13] B/D Select(1-8), RD, WR 신호 발생



[그림 14] B/D Select(1-8), 제어신호(A2~A5)발생



[그림 15] 채널1개 읽기 수행 후 다음 채널 읽기 수행 주기

그림 14는 해당 채널에 대한 제어신호 A2~A5 (Lab8~Lab11) 발생을 보여주며 그림 15는 1채널에 대한 읽기 수행 후 다음 채널 읽기 수행까지의 주기를 보여 주며 모두 정상 동작함을 확인하였다.

또한, 각 채널별, 보드별로 분석한 데이터를 바탕으로 전체 수행 처리시간 및 주기를 살펴았다. 1개의 노드에 대한 읽기 처리 시간()은 2.8us이고 18개 보드에 대한 읽기 수행시간()은 기타 제어시간()까지 포함하면 식 3과 같이 362us가 걸림을 알 수 있다. 또한, 노드 1개에 대한 쓰기 후에 읽기 작업은 3채널에 대해 모두 수행되어야 하므로 식 2과 같이 10.086ms가 걸리며 1개의 보드에 대한 쓰기/읽기 수행시간()은 식 3과 같이 약 1초, 18개 입 · 출력 버퍼 처리장치 모두

에 대해서는 약 18초 정도 소요되는 것을 확인하였다.

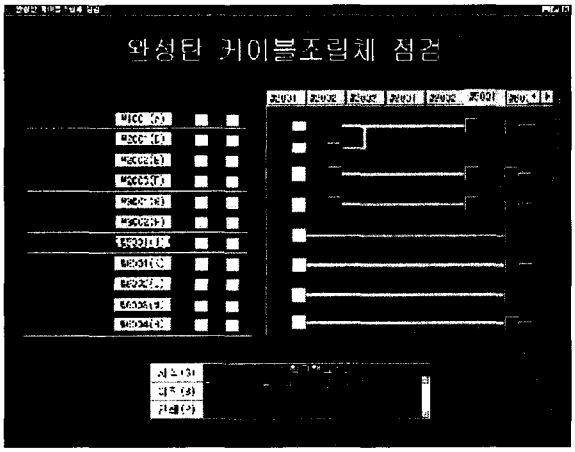
$$t_R = n_{board} \cdot t_n + t_{other} = 362\mu s, \quad (3)$$

$$T_{WR} = t_W + n_{ch} \cdot t_R = 10.086ms \quad (4)$$

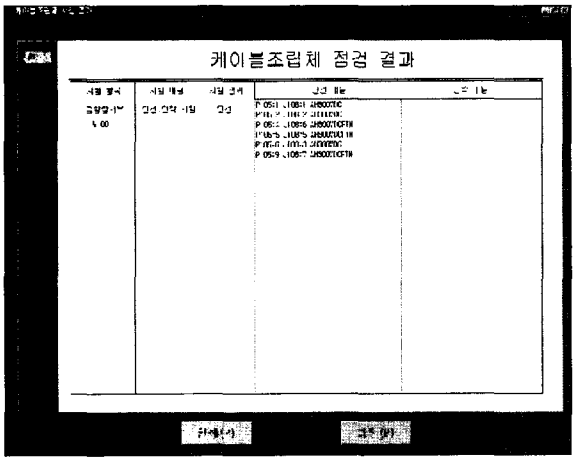
$$T_{board} = T_{wr} \cdot n_{bit} \cdot n_{ch} = 998.8128ms \quad (5)$$

(단, n_{board} = 18, n_{ch}(채널개수) = 3, n_{bit}(채널폭) = 32)

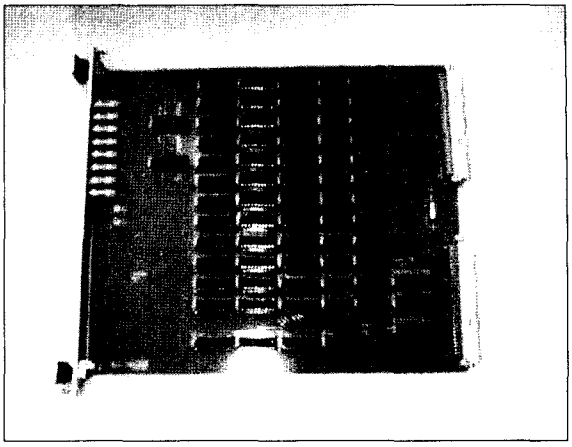
5.3 클라이언트 데이터 분석



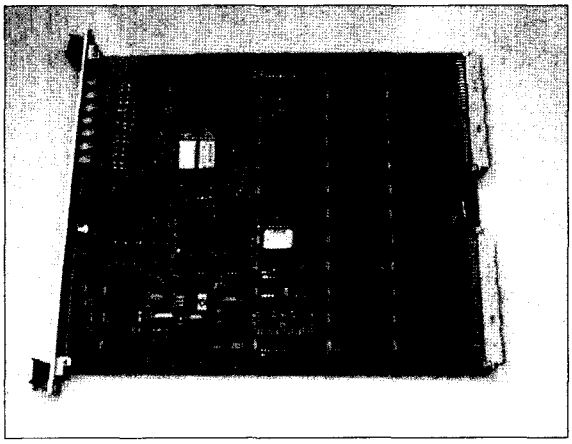
[그림 16] 점검 케이블 선택 화면



[그림 17] 케이블 점검 결과 확인



[그림 18] 구현된 입 · 출력 버퍼 처리장치 사진



[그림 19] 구현된 주처리장치 사진

클라이언트는 점검하고자 하는 케이블을 선택하고 그 점검 결과를 GUI 상태로 보여 주는 역할을 수행한다. 케이블에 대한 정보는 파일로 저장되어 클라이언트가 해당 케이블 점검 수행 시에 그 정보를 참조하도록 하였으므로 해당 케이블에 대한 정보가 변경되더라도 쉽게 수정, 편집이 가능하다. 클라이언트에 대한 점검 결과 분석은 해당 케이블에 대하여 단선, 단락 결과가 정상으로 출력되는지의 여부를 확인하였으며 또한, 일부러 잘못된 케이블 또는 케이블 정보 입력으로 그 결과가 나타나는 지를 확인하였다. 그림

16은 점검 케이블을 선택하는 그림이며 그림 17은 일부러 단선을 수행한 점검 결과를 보여준다. 점검하고자 하는 케이블에 대한 모든 점검을 수행하였으며 그 결과가 정상적으로 수행됨을 확인하였다.

그림 18과 그림 19는 각각 VME 방식으로 구현된 입 · 출력 처리장치 및 주처리장치 사진이다.

6. 결 론

본 논문에서는 기존의 릴레이 매트릭스 방식이 대용량 케이블의 점검에 적합하지 않다는 단점을 보완하여 새로운 입 · 출력 버퍼방식을 이용한 대용량 케이블 점검 시스템을 설계 및 구현하였다.

케이블 설계 변경이 잦은 시스템에 적용가능하도록 32비트 3채널로 구성된 입 · 출력 버퍼 처리장치를 VME 방식의 보드로 모듈화하여 가변적인 구성이 용이하도록 하였다. 케이블 연결 정보는 모듈화된 설계에 적합하도록 편집이 용이한 DB로 구축하였다.

또한, 18개의 입 · 출력 처리장치를 제어하는 마이크로 프로세서(MPC860) 기반의 주처리장치 설계로 1728개 노드에 대한 총 점검 시간이 보드당 1초 이내로 고속 점검이 가능함을 확인하였다.

본 시스템 설계 내용은 케이블 점검을 필요로 하는 임의의 대용량 케이블 점검에 대하여 효율적인 적용이 가능하다.

참 고 문 헌

[1] 백문흠 외, "현무 유도탄 케이블 시험셋트 최종 연구 보고서", 국방과학연구소 NSRD-409-89206, 1989. 7.

[2] McConnell, A., "An Automatic Cable Checker ; a fourth generation device", IEEE Proceedings of, 1991. vol.1, pp.23～27.

[3] 양종원 외, "청상어 케이블 점검장비 개발" , 국방과학연구소 NWSD-519-010083, 2001. 1.

[4] Schaelicke,L., "Improving I/O Perfo- rmance with a Conditional Store buffer.", 31st Annual ACM/IEEE International Symposium, 1999, pp.160～169.

[5] Secareanu, R.M., "Low Power Digital CMOS Buffer Systems for Driving Highly Capacitive Interconnect Lines.", Proceedings of the 43rd IEEE Midwest Symposium on, Vol.1, 2000, pp.362～365.

[6] 설병수 외, "대용량 Dynamic RAM의 Data Retention 테스트 회로 설계", 전자공학회논문지, 30-A권 9호, 1993년 9월, pp.736～747.

[7] 김영로 외, "클럭주기 최소화를 위한 효율적인 연결구조 할당 알고리즘", 전자공학회논문지, 32-A권 6호, 1995년 6월, pp.849～861.

[8] Motorola Inc., MPC860 PowerQUICC User's Manual, July. 1998, pp.1-1～16-78.

[9] Motorola Inc., PowerPC Micro- processor Family : The Bus Interface for 32-Bit Microprocessors, 1997.

[10] Jen-Chieh Tuan, "On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture.", IEEE Trans. on Circuit and System. vol.12, No.1, Jan. 2002, pp.61～72.